Style Mining of Electronic Messages for Multiple Authorship Discrimination: First Results

Shlomo Argamon
Dept. of Computer Science
Illinois Institute of Technology
10 W. 31st Street
Chicago, IL
argamon@iit.edu

Marin Šarić*
Dept. of Computer Science
Illinois Institute of Technology
10 W. 31st Street
Chicago, IL
marin@google.com

Sterling S. Stein
Dept. of Computer Science
Illinois Institute of Technology
10 W. 31st Street
Chicago, IL
stein@ir.iit.edu

ABSTRACT

This paper considers the use of computational stylistics for performing authorship attribution of electronic messages, addressing categorization problems with as many as 20 different classes (authors). Effective stylistic characterization of text is potentially useful for a variety of tasks, as language style contains cues regarding the authorship, purpose, and mood of the text, all of which would be useful adjuncts to information retrieval or knowledge-management tasks. We focus here on the problem of determining the author of an anonymous message, based only on the message text. Several multiclass variants of the Winnow algorithm were applied to a vector representation of the message texts to learn models for discriminating different authors. We present results comparing the classification accuracy of the different approaches. The results show that stylistic models can be accurately learned to determine an author's identity.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.3.3 [**Information Search and Retrieval**]: Selection process

Keywords

Text mining, Text categorization, Authorship attribution, Computational stylistics, Electronic communication

1. INTRODUCTION

As electronic communication increasingly occupies a central role in the global business, political, and intelligence communities, it has become increasingly important to be able to accurately identify the authors of an electronic messages. For example, it may be important to know which intercepted documents discuss terrorist attack plans, but it is equally important to pick out those messages known to be from major enemy players. However, a message author who wishes to remain unidentified currently has many ways to do so.

We consider the use of computational stylistics for authorship attribution of electronic messages, addressing categorization problems with as many as 20 different classes (authors). We compare results using several multi-class generalizations of the Exponentiated Gradient learning algorithm (EG); we have shown in previous work [12] that the EG algorithm outperforms other popular learning methods for stylistic classification.

Author attribution of electronic messages constitutes an important and difficult testbed for computational stylistics techniques, for two main reasons. First is the fact that such messages are typically quite short (tens or perhaps hundreds of words compared with thousands for articles or books). Thus there is less information in any given text to go on. Second, since e-mail is an informal and fast-paced medium, individuals' writing styles adapt quickly to different contexts or correspondents, and so even messages by the same individual may vary greatly in style.

The usual goal in text mining and analysis is to gain an understanding or summary of the topic, or topics, covered in the text (see [17] for an excellent survey). However, the meaning of a text is more than the topic it describes or represents. Textual meaning, broadly construed, includes also dimensions of: *affect* (what feeling is conveyed by the text?), *genre* (in what community of discourse does the text function?), *register* (what is the function of the text as a whole?), and *personality* (what sort of person, or who, wrote the text?). These aspects of meaning are largely captured by the text's *style* of writing, which may be roughly defined as how the author chose to express her topic, from among a very large space of possible ways of doing so. This contrasts, thus, the **how** of a text (style) from the **what** (topic).

Due to the focus on topic-based text analysis, little work to date has been done on text mining using style. However, while topic is the most easily accessible aspect of a text's meaning, since content words carry much of a text's topical meaning, the complementary problem of understanding textual style is also crucial to understanding a text in its context. Therefore stylistic text analysis is a critical component of effective processing for the growing and increasingly complex text collections available today.

*Current affiliation: Google, Inc., 1505 Salado Drive, Mountain View, CA 94043

SIGKDD'03, August 24-27, 2003, Washington, DC, USA

2. COMPUTATIONAL STYLISTICS

The problem considered in this paper concerns categorization by style and thus may be viewed in the context of the stylometric research which has been vigorously pursued for decades, mostly in the context of authorship attribution [9, 14]. Although some crossover work between stylometrics and text categorization has been done, the bulk of stylometric research has differed from the more recent work in text categorization in a few important ways.

While categorization by topic is typically based on keywords which reflect a document's content, categorization by author style uses precisely those features which are independent of content. Most generally, style is carried by features that indicate the author's choice of one mode of expression from among a set of equivalent modes. This may be expressed through lexical choice, choice of syntactic structure, or of discourse strategy.

In order to practically approach the problem, stylometric models for categorization have typically been based on hand-selected sets of content-independent, lexical [16], syntactic [18], or complexity-based [21] features. Researchers in text categorization by topic typically use much larger feature sets, often in conjunction with automated feature selection methods; some work in the stylometric community has also considered automated methods for selecting features [7]. Moreover, stylometric research has tended to use statistical methods such as multivariate analysis [2], rather than machine-learning algorithms, for categorization.

Computational stylistics comprises a growing body of work in which machine learning techniques are used to address problems of document stylistic analysis. Multilayer perceptron networks were applied to frequencies of a small set of function words as features by Matthews and Merriam [13] and Tweedie et al. [19]. Wolters and Kirsten [20] compared nearest-neighbor learning algorithms with part-of-speech tag frequencies and demonstrated effective classification based on genre. More complex syntactic features were used by Stamatatos et al. [18] in conjunction with a discriminant analysis to classify Greek newspaper articles by genre. Argamon et al. [1] have used the EG algorithm [11] to learn classification models for distinguishing male from female authorship, using function words and shallow syntactic features. The problem of authorship identification for e-mail has been previously addressed by de Vel [6], who used a comparatively small number of features with SVMs [4] for binary authorship classification. His results indicate (inter alia) the potential importance of orthographic text features, such as capitalization and word-length, in style analysis.

3. THE CORPUS

Unfortunately, there are not yet any publically available benchmark collections for experimentation on author attribution of electronic messages. We have therefore built such a collection, which we are making available to the research community at large. Due to privacy considerations, we could not use a corpus of personal e-mail messages. Instead, we used messages culled from several Usenet newsgroups[1] on a variety of topics. The groups we chose are not edited, and hence individual messages may drift from time to time from the nominal topic, though we expect the topic of the group to set a general tenor. For this initial work, we chose several newsgroups on a range of topics: `rec.arts.books` (abbreviated `books`), `comp.theory` (`theory`), and `comp.lang.c` (`lang.c`). Posts in `comp.lang.c` often contain code snippets which we could not easily reliably filter out.

[1]See `http://groups.google.com`.

Table 1: Summary of statistics of the Usenet post corpora used in the study: maximum, minimum, and average number of posts for any author, percentage of posts by the maximum author, and maximum, minimum, and average length (in word tokens) of each post.

Subcorpus	posts/author min/max/avg	max %	tokens/post min/max/avg
books(2)	116/131/124	53%	9/409/61
books(5)	89/131/111	24%	9/1250/95
books(20)	33/131/67	10%	3/4142/102
books(10/)	10/131/52	13%	3/1250/85
theory(2)	84/102/93	55%	12/584/167
theory(5)	51/102/79	26%	11/584/155
theory(20)	15/102/40	13%	6/3111/150
theory(10/)	10/102/34	15%	11/3111/151
lang.c(2)	162/300/231	65%	12/287/85
lang.c(5)	128/300/175	34%	7/399/75
lang.c(20)	55/300/101	15%	7/445/79
lang.c(10/)	10/300/74	20%	7/445/79

From each newsgroup we downloaded the 500 most recent discussions (*threads*), each thread consisting of many individual postings. Each posting was then processed to remove any material quoted from previous posts as well as extraneous header material, the newsgroup and author of the post was recorded. Each set of newsgroup posts was then used to compose several subcorpora with different numbers of authors for attribution: all posts by the (i) 2 most frequent authors, (ii) 5 most frequent authors, (iii) 20 most frequent authors, and (iv) 10 most frequent and 10 least frequent authors (minimum of 10 posts each in the corpus), denoted "10/". Table 1 gives details regarding the sizes of these subcorpora.

4. TEXT FEATURES

We use here lexical/orthographic features as indicator variables for stylistic choice at multiple levels of linguistic structure. The kinds of features that may be reliable stylistic indicators are somewhat dependent on the domain of text being considered. The corpus considered in this study consists of informal messages in a variety of Internet newsgroups, a medium which allows fairly wide stylistic latitude, but which also has its own conventions (observed by different individuals to differing extents). Thus the features that we considered are a combination of generic and newsgroup-specific ones.

Most basic is a list of 303 generic function words (taken from [15][2]), which generally serve as proxies for choice in the syntactic (e.g., preposition phrase modifiers vs. adjectives or adverbs), semantic (e.g., usage of passive voice indicated by auxiliary verbs), and pragmatic (e.g., first-person pronouns indicating personalization of a text) planes. Function words have been shown to be effective style markers in previous

[2]Available on-line at `http://www.cse.unsw.edu.au/~min/ILLDATA/Function_word.htm`

studies [1, 10, 19]. We add to the list of function words also lexical items special to the newsgroup domain: *net abbreviations* (netabbrevs). Netabbrevs are typically acronyms for commonly used locutions, such as "BTW" (by the way) or "N2S" (needless to say). Informal observation suggests that different user communities often use somewhat different sets of netabbrevs. We took as features a list of 190 netabbrevs taken from a webpage listing[3].

In addition, we also considered a parallel set of orthographic and placement features, both individually and in combination with the above lexical features (where applicable): Capitalization (Uppercase, lowercase, ALL UPPERCASE, MiXeD CaSe, punctuation, number), Placement (Subject, beginning of line, beginning of paragraph, other), Word length, and Line length.

5. LEARNING ALGORITHMS

The primary task we address here is to learn a model to classify electronic messages according to their author, based on stylistic features as above. We examine here multiclass variants of the Exponentiated Gradient (EG) algorithm [11]. The EG algorithm has nice theoretical mistake-bound properties and variants have previously been shown to be effective for text-categorization by topic (e.g. [5]). In fact, in our previous work on stylistic text categorization [12], a variant of EG considerably outperformed both Naive Bayes and Ripper, other algorithms commonly used for text categorization.

5.1 Multiplicative-update learning

The basic EG algorithm learns linear classifiers for two-class problems. Briefly, the balanced EG variant upon which we base our work is as follows. We initialize two component weight vectors $w^+ = \{1, 1, \ldots, 1\}$ and $w^- = \{-1, \ldots, -1\}$, defining $w = w^+ + w^-$. We then calibrate the vectors using the following iterative procedure. The training examples are randomly ordered. For each training example x, we define $c(x) \in \{0, 1\}$ to be the class indicator function. Let $s(w, x) = 1$ if $w \cdot x > 0$ and $s(w, x) = 0$ otherwise, where w is the weight vector at the time that example x is encountered, and let w_i and x_i be the ith element in w and x, respectively. Then we take the examples one at a time in a random order and iteratively update the weights after each example using the formulas:

$$\begin{aligned} w_i^+ &\leftarrow w_i^+(1+\beta)^{x_i(c(x)-s(w,x))} \\ w_i^- &\leftarrow w_i^-(1+\beta)^{x_i(s(w,x)-c(x))} \end{aligned}$$

β is a learning constant greater than 0; in all our experiments we used $\beta = 2$. Thus weights that improperly reduce the dot product are increased, and vice versa. If $(c(x)-s(w,x)) > 0$, and so w^+ is increased, the update is termed *promotion*, and conversely if $(c(x) - s(w, x)) < 0$ the update is termed *demotion*. Note that as in EG, but unlike Balanced Winnow, we allow x_i to take on non-binary values. However, like Balanced Winnow, but unlike EG, we restrict $s(w, x)$ to binary values. A computationally similar variant of this update rule which we have found to work somewhat better than the above rule for stylistic classification, is:

$$\begin{aligned} w_i^+ &\leftarrow w_i^+(1+\beta x_i)^{(c(x)-s(w,x))} \\ w_i^- &\leftarrow w_i^-(1+\beta x_i)^{(s(w,x)-c(x))} \end{aligned}$$

[3] `http://www.geocities.com/TheTropics/Shores/1224/Netiquette.html`

Once all the examples have been used for training, they are randomly reordered and another cycle of updates is run. (Convergence is usually reached within 20 or 30 iterations.) Along the way, any element of w^+ or w^- that drops below some threshold (0.000001 of the sum of all the weights) is set to zero.

The intuition behind the update rule is that weights of features that appear most prominently in misclassified documents are changed most dramatically. A well-known advantage of multiplicative update rules such as we use is that the weights of irrelevant features tend quickly to zero. This is important for problems such as stylistic classification, where determining relevant features in advance is particularly difficult.

5.2 Multiclass learning

The usual way to extend a two-class learning algorithm for multiclass problems is to learn a set of two-class models, and then to combine their binary classifications and confidences thereof to give a final class for an example. This combination is usually performed using a type of "winner-take-all" strategy. A widely-used paradigm is *one vs. all* (OvA), in which, for a learning problem with n classes $c_1, \ldots, c_n$, the system learns n models for the binary problems: "c_1 vs. everything else", "c_2 vs. everything else", and so on. Learning will thus output an array M of models, M_1 through M_n. Then, to classify a test example x, that class is chosen whose model gives the highest confidence in its classification (for linear classifiers, confidence for model m on example x is easily measured as $m \cdot x$). While this method is reasonably efficient, it can be shown that there are concepts that cannot be represented by OvA models, even though they can be learned by ensembles of linear classifiers.

A second combination method is *all vs. all* (AvA) in which $n(n-1)$ models are learned, for the problems "c_1 vs. c_2", "c_1 vs. c_3", and so on through "c_{n-1} vs. c_n". Here the output will be a 2-dimensional array of models (with empty diagonal), $M_{i,j}$, $i, j \leq n, i \neq j$. To classify a test example x, then, for each possible class c_i, the confidences of the $n-1$ classifiers indicating c_i are combined, usually by summing them (weighted voting). We consider here also two other variants: using the minimum confidence for a class, and using the maximum confidence for a class. Once the combined confidence for each class is computed, the class with the highest combined confidence is chosen.

The naive implementation of the above schemes (given in Fig. 2) has each component model calibrating its weight vector independently of all other component models. However, this can lead to inefficiencies and even inaccuracies in learning, as the final classification may depend on interactions between the component models. A recent technique which addresses this issue has been termed "ultraconservative algorithms" [3]. Consider that, in the multi-class case, a given component model M_i is only "correct" or "incorrect" for a given training example $(x, c(x))$ in relation to the confidences given by the other component models for that example. For the OvA case, we have then:

$$\begin{aligned} M_{i=c(x)} \text{ is correct} \quad &\text{iff} \quad \forall j \neq i,\ M_i \cdot x > M_j \cdot x \\ M_{i \neq c(x)} \text{ is correct} \quad &\text{iff} \quad M_i \cdot x \leq M_{c_x} \cdot x \end{aligned}$$

Thus all models for an incorrect class with confidence higher than the model for the correct class $c(x)$ should be demoted for x, and the model for the correct class should be promoted

MULTICLASSLEARN($\mathcal{X}, c, n_{iter}$):

1. Initialize model array M (variant-dependent)
2. For $i \leftarrow 0$ to n_{iter}, do:
   (a) Reorder $\mathcal{X}$ randomly
   (b) For each $x \in \mathcal{X}$, UPDATE($x, c(x), M$)
   (c) For each $m \in M$, NORMALIZE(m)
3. Output M

NORMALIZE(m):

1. Let $m = (w^+, w^-)$
2. For each i:
   (a) $w_i^+ \leftarrow w_i^+ / (\sum_i w_i^+)$
   (b) $w_i^- \leftarrow w_i^- / (\sum_i w_i^-)$

Figure 1: Generic update-based multi-class learning algorithm for example set $\mathcal{X}$, class indicator function c, and number of iterations n_{iter}. Regarding UPDATE, see text and Figures 2 and 3.

NAIVEONEVSALLUPDATE(x, c, M_n):

1. For each $i \leq n$:
   (a) If $i \neq c$ and $M_i \cdot x > 0$, then: DEMOTE(M_i, x)
   (b) If $i = c$ and $M_i \cdot x \leq 0$, then: PROMOTE(M_i, x)

NAIVEALLVSALLUPDATE(x, c, M_n):

1. For each $i \leq n$, $i \neq c$:
   (a) If $M_{i,c} \cdot x > 0$, then: DEMOTE($M_{i,c}, x$)
   (b) If $M_{c,i} \cdot x \leq 0$, then: PROMOTE($M_{c,i}, x$)

Figure 2: Naive update variants for example x, correct class c, and model array M. See text for details.

for x exactly when it does not give the highest confidence.

This schema we implement in three variants, one for the OvA classification paradigm, and two for the AvA paradigm (refer to Fig. 3). For ONEVSALL update, we compare the model confidence for each incorrect class to that for the correct class. If the incorrect class scores higher than the correct class, we update both models appropriately. In the straightforward update for the AvA case, which we call ALLVSALL update, we compare all models $M_{c,j}$ whose confidence indicates the correct class c over some other class j to all models $M_{i,c}$ whose confidence indicates an incorrect class i over the correct class c. For each such pair-wise comparison, if the 'incorrect' model has a higher confidence than the 'correct' model, the models are appropriately updated. This 'model-by-model' updating rule, however, does not take into account the fact that model confidences are combined in the eventual classification. Hence we also propose the ROWVSROW update rule, in which the combined confidence for each incorrect class is compared to the combined confidence for the correct class. If the incorrect class's confidence is higher,

ONEVSALLUPDATE(x, c, M_n):

1. For each $i \leq n$, $i \neq c$:
   (a) If $M_i \cdot x \geq M_c \cdot x$, then:
      i. DEMOTE(M_i, x)
      ii. PROMOTE(M_c, x)

ALLVSALLUPDATE($x, c, M_{n,n}$):

1. For each $i, j \leq n$, $i, j \neq c$:
   (a) If $M_{i,c} \cdot x \geq M_{c,j} \cdot x$, then:
      i. DEMOTE($M_{i,c}, x$)
      ii. PROMOTE($M_{c,j}, x$)

ROWVSROWUPDATE($x, c, M_{n,n}$):

1. For each $i \leq n$, $i \neq c$:
   (a) If $\sum_j M_{i,j} \cdot x \geq \sum_j M_{c,j} \cdot x$, then:
      i. For each $j \leq n$, DEMOTE($M_{i,j}, x$)
      ii. For each $j \leq n$, PROMOTE($M_{c,j}, x$)

Figure 3: Ultraconservative update variants for example x, correct class c, and model array M. See text for details.

then all models in both rows are updated. Details of the update algorithms are given in Fig. 3.

In order to ensure proper convergence, balance must be maintained between the various weight vectors. In the present work we use a simple method of normalizing the weight vectors after each pass through the training set (see Fig. 1). While we have as yet no convergence proof, our experiments (below) show that this normalization scheme gives reliable convergence.

6. EXPERIMENTS

6.1 Setup

For each subcorpus of a newsgroup and a selection of authors therefrom, we evaluated the authorship attribution performance of different learning methods using 10-fold cross-validation. We measured classification accuracy, and in addition examined the precision-recall behavior by examining average $F_{\beta=1}$ over the individual authors in each dataset, as well as average precision over the authors in each dataset. Comparing these numbers allows us to roughly evaluate the stability of each method over the different authors in each dataset.

The primary choice of learning method is between "one vs. all" (OvA) and "all vs. all" (AvA). Within OvA there are two variants, the naive and the ultraconservative (denoted "Cons" in the tables). Within AvA, there are three combination methods (vote, min, and max), and three modes of ultraconservative update (none, all model pairs, and by rows). Results are given in Tables 2, 3, 4, and 5.

6.2 Classification Results

Due to the lack of a standard testbed for author attribution in this domain, we evaluate efficacy of the approached

Table 2: Summary of classification results for One vs. All learning, with overall accuracy, min, max, and average $F_{\beta=1}$ with standard deviation, and average precision with standard deviation. 'Group' is the newsgroup. 'N' is the number of authors, either the 2, 5, or 20 most frequent or 10/, the 10 most and least frequent authors. 'Con' is whether or not learning was ultraconservative.

Group	N	Con	Acc	F_{min}	F_{max}	F_μ	P_μ
books	2	no	66%	0.65	0.67	0.66	0.67
books	2	yes	67%	0.63	0.71	0.67	0.70
books	5	no	40%	0.29	0.52	0.39	0.42
books	5	yes	43%	0.21	0.56	0.41	0.45
books	10/	no	22%	0 (4)	0.44	0.17	0.26
books	10/	yes	21%	0 (4)	0.44	0.16	0.24
books	20	no	20%	0.03	0.55	0.19	0.24
books	20	yes	24%	0.10	0.60	0.24	0.29
theory	2	no	73%	0.69	0.76	0.73	0.73
theory	2	yes	70%	0.65	0.74	0.70	0.70
theory	5	no	42%	0.24	0.77	0.43	0.56
theory	5	yes	41%	0.27	0.78	0.45	0.40
theory	10/	no	18%	0 (12)	0.40	0.066	0.13
theory	10/	yes	29%	0 (5)	0.74	0.19	0.29
theory	20	no	19%	0 (8)	0.42	0.096	0.17
theory	20	yes	26%	0 (4)	0.78	0.19	0.27
lang.c	2	no	99%	0.99	0.99	0.99	0.99
lang.c	2	yes	99%	0.99	0.99	0.99	0.99
lang.c	5	no	73%	0.46	0.96	0.68	0.78
lang.c	5	yes	68%	0.33	0.95	0.61	0.67
lang.c	10/	no	47%	0 (4)	0.85	0.25	0.35
lang.c	10/	yes	53%	0 (5)	0.89	0.31	0.34
lang.c	20	no	33%	0.026	0.72	0.23	0.40
lang.c	20	yes	43%	0 (1)	0.94	0.30	0.76

described here both comparatively, and against the simplest baseline of always classifying according to the majority class in each dataset (see Table 1).

We first examine OvA learning, in Table 2. First note that in all cases, we obtained classification accuracies noticeably higher than the baseline, establishing the validity of the method. Furthermore, in most cases ultraconservative learning gives a slight improvement, and never significantly reduces accuracy. This empirically confirms the usefulness of ultraconservative updating. Significantly, all metrics examined give qualitatively similar results. (Note that for two-classes, all algorithms examined are theoretically identical, so some results are omitted due to lack of space.)

If results for AvA learning are now compared to those for OvA, a distinct pattern emerges which is consistent in all datasets. Apparently, "ultraconservative update by rows" is usually preferred to other AvA variants as well as to OvA learning, usually with either voting or maximum combination. These strategies sometimes give results nearly 30% higher than the baseline. Also, the $F_{\beta=1}$ range for these variants is usually narrow, indicating both overall reliability and consistency for each author. To our knowledge, the row-based variant of ultraconservative updating has not previously been examined, so it bears further scrutiny (perhaps in the framework of "constraint classification" [8]).

Table 3: Summary of classification results for All vs. All learning, for newsgroup rec.arts.books. 'Com' is the combination method: 'vote', 'min', or 'max'. 'Con' is how ultraconservative learning was applied: 'none', 'all', or 'by rows'.

N	Com	Con	Acc	F_{min}	F_{max}	F_μ	P_μ
5	vote	none	38%	0.29	0.44	0.35	0.46
5	vote	all	33%	0.18	0.43	0.30	0.41
5	vote	rows	46%	0.38	0.58	0.46	0.48
5	min	none	40%	0.32	0.49	0.38	0.45
5	min	all	36%	0.25	0.44	0.34	0.43
5	min	rows	34%	0.13	0.48	0.29	0.37
5	max	none	29%	0.14	0.41	0.23	0.33
5	max	all	29%	0.11	0.40	0.24	0.35
5	max	rows	45%	0.35	0.51	0.46	0.46
10/	vote	none	30%	0 (10)	0.46	0.13	0.12
10/	vote	all	23%	0 (6)	0.38	0.15	0.22
10/	vote	rows	21%	0 (6)	0.35	0.14	0.23
10/	min	none	24%	0 (3)	0.43	0.17	0.20
10/	min	all	25%	0 (4)	0.52	0.18	0.24
10/	min	rows	16%	0 (7)	0.27	0.10	0.17
10/	max	none	17%	0 (10)	0.24	0.082	0.086
10/	max	all	14%	0 (10)	0.25	0.071	0.073
10/	max	rows	23%	0 (4)	0.37	0.17	0.21
20	vote	none	25%	0 (4)	0.47	0.18	0.26
20	vote	all	21%	0 (2)	0.35	0.17	0.30
20	vote	rows	23%	0 (2)	0.56	0.21	0.25
20	min	none	26%	0.08	0.41	0.24	0.30
20	min	all	23%	0 (2)	0.55	0.20	0.28
20	min	rows	14%	0 (5)	0.26	0.091	0.23
20	max	none	11%	0 (6)	0.28	0.072	0.098
20	max	all	9.5%	0 (5)	0.20	0.067	0.11
20	max	rows	24%	0 (1)	0.47	0.18	0.27

Table 4: Summary of classification results for All vs. All learning, for comp.theory; columns as in Table 3.

N	Com	Con	Acc	F_{min}	F_{max}	F_μ	P_μ
5	vote	none	43%	0.28	0.67	0.44	0.55
5	vote	all	37%	0.25	0.66	0.38	0.50
5	vote	rows	41%	0.27	0.86	0.43	0.55
5	min	none	40%	0.27	0.64	0.42	0.51
5	min	all	39%	0.26	0.73	0.41	0.47
5	min	rows	39%	0.15	0.47	0.36	0.44
5	max	none	31%	0.10	0.52	0.31	0.44
5	max	all	28%	0.16	0.44	0.26	0.41
5	max	rows	43%	0.24	0.83	0.44	0.51
10/	vote	none	28%	0 (11)	0.57	0.11	0.13
10/	vote	all	29%	0 (7)	0.67	0.17	0.31
10/	vote	rows	23%	0 (4)	0.67	0.17	0.27
10/	min	none	24%	0 (2)	0.61	0.16	0.25
10/	min	all	21%	0 (5)	0.60	0.14	0.20
10/	min	rows	18%	0 (12)	0.25	0.072	0.094
10/	max	none	17%	0 (11)	0.31	0.063	0.096
10/	max	all	11%	0 (10)	0.23	0.050	0.070
10/	max	rows	26%	0 (6)	0.70	0.18	0.23
20	vote	none	24%	0 (8)	0.67	0.17	0.21
20	vote	all	20%	0 (9)	0.34	0.10	0.20
20	vote	rows	24%	0 (3)	0.74	0.20	0.24
20	min	none	23%	0 (3)	0.35	0.15	0.24
20	min	all	20%	0 (8)	0.46	0.11	0.20
20	min	rows	15%	0 (16)[4]	0.24	0.03	0.11
20	max	none	13%	0 (8)	0.39	0.075	0.12
20	max	all	11%	0 (10)	0.27	0.056	0.080
20	max	rows	22%	0 (2)	0.73	0.18	0.23

Table 5: Summary of classification results for All vs. All learning, for `comp.lang.c`; columns as in Table 3.

N	Com	Con	Acc	F_{min}	F_{max}	F_μ	P_μ
5	vote	none	65%	0.45	0.88	0.62	0.67
5	vote	all	62%	0.42	0.83	0.60	0.68
5	vote	rows	64%	0.47	0.89	0.59	0.67
5	min	none	71%	0.42	0.96	0.65	0.71
5	min	all	66%	0.26	0.95	0.58	0.68
5	min	rows	62%	0.32	0.90	0.58	0.68
5	max	none	48%	0.32	0.76	0.45	0.58
5	max	all	35%	0.32	0.39	0.36	0.58
5	max	rows	59%	0.26	0.90	0.53	0.66
10/	vote	none	43%	0 (10)	0.85	0.19	0.25
10/	vote	all	29%	0 (9)	0.47	0.15	0.21
10/	vote	rows	55%	0 (2)	0.89	0.35	0.40
10/	min	none	47%	0 (2)	0.94	0.29	0.33
10/	min	all	46%	0 (3)	0.92	0.29	0.34
10/	min	rows	47%	0 (5)	0.83	0.24	0.27
10/	max	none	20%	0 (10)	0.31	0.093	0.13
10/	max	all	16%	0 (9)	0.57	0.093	0.11
10/	max	rows	53%	0 (1)	0.84	0.32	0.38
20	vote	none	29%	0 (1)	0.75	0.22	0.38
20	vote	all	27%	0 (2)	0.80	0.25	0.39
20	vote	rows	45%	0.063	0.87	0.38	0.42
20	min	none	43%	0.042	0.88	0.37	0.41
20	min	all	35%	0 (1)	0.89	0.29	0.38
20	min	rows	40%	0 (1)	0.83	0.31	0.33
20	max	none	13%	0 (4)	0.28	0.086	0.20
20	max	all	11%	0 (3)	0.31	0.086	0.23
20	max	rows	39%	0.028	0.83	0.30	0.34

7. CONCLUSIONS

This paper presents initial results for authorship attribution on electronic messages, using linear separator learning and lexical/orthographic features. We tested a variety of learning algorithms based on EG, using a test corpus which we are making publically available. It will constitute a first benchmark collection for research on stylistic attribution.

Generally speaking, our results are quite promising and bear witness to the validity of the approach. The major limitation of the current work is the simplicity of its feature set—though large, the set of features considered here is comparatively simple. Future work will include expanding the feature set to include other types of style markers, such as parts-of-speech and complexity metrics. As well, we intend to investigate the use of automatic feature generation techniques to search the space of compound features in order to improve learning accuracy. We believe that this will be necessary, due to the unavoidable idiosyncracy of style.

Among other methods for multiclass linear separator learning, we presented the novel technique of "ultraconservative updating by rows". Our results suggest the possible preference of this new technique for electronic message authorship attribution. The technique therefore merits further study, both for authorship attribution and for other applications.

Acknowledgements

Thanks to Ophir Frieder and Steven Beitzel for their careful readings of early drafts of this paper, as well as to the anonymous reviewers for their helpful comments.

8. REFERENCES

[1] S. Argamon, M. Koppel, J. Fine, and A. R. Shimony. Gender, genre, and writing style in formal written texts. *Text*, 23(3), 2003.

[2] J. F. Burrows. Computers and the study of literature. In *Computers and Written Texts*, pages 167–204. Oxford: Blackwell, 1992.

[3] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. In *Proc. COLT/EuroCOLT*, pages 99–115, Amsterdam, 2001.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction To Support Vector Machines*. Cambridge U. Press, 2000.

[5] I. Dagan, Y. Karov, and D. Roth. Mistake-driven learning in text categorization. In *Proc. EMNLP-97*, Providence, RI.

[6] O. de Vel. Mining e-mail authorship. In *KDD-2000 Workshop on Text Mining*, Boston, MA, 2000.

[7] R. S. Forsyth and D. I. Holmes. Feature finding for text classification. *Lit. and Ling. Comp.*, 11(4):163–174, 1996.

[8] S. Har-Peled, D. Roth, and D. Zimak. Constraint classification for multiclass classification and ranking. In *NIPS-15*, 2002.

[9] D. I. Holmes. The evolution of stylometry in humanities scholarship. *Lit. and Ling. Comp.*, 13(3):111–117, 1998.

[10] J. Karlgren. *Stylistic Experiments for Information Retrieval*. PhD thesis, SICS, 2000.

[11] J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

[12] M. Koppel, S. Argamon, and A. R. Shimoni. Automatically categorizing written texts by author gender. *Lit. and Ling. Comp.*, 17(4), 2003.

[13] R. A. J. Matthews and T. V. N. Merriam. Neural computation in stylometry I: An application to the works of Shakespeare and Fletcher. *Lit. and Ling. Comp.*, 8:103–209, 1993.

[14] A. McEnery and M. Oakes. *Authorship studies/textual statistics*, pages 234–248. Marcel Dekker, 2000.

[15] R. Mitton. Spelling checkers, spelling correctors and the misspellings of poor spellers. *Information Processing and Management*, 23(5):495–505, 1987.

[16] F. Mosteller and D. Wallace. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley, Reading, Massachusetts, 1964.

[17] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 2002.

[18] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Automatic text categorisation in terms of genre and author. *Comp. Ling.*, 26(4):471–495, 2001.

[19] F. Tweedie, S. Singh, and D. Holmes. Neural network applications in stylometry: The federalist papers. *Computers and the Humanities*, 30(1):1–10, 1996.

[20] M. Wolters and M. Kirsten. Exploring the use of linguistic features in domain and genre classication. In *Proc. EACL '99*, pages 142–149, 1999.

[21] G. U. Yule. *Statistical Study of Literary Vocabulary*. Cambridge U. Press, 1944.